LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-643477

Matrix Reordering Using Multilevel Graph Coarsening for ILU Preconditioning

D. Osei-Kuffuor, R. Li, Y. Saad

September 5, 2013

Matrix Reordering Using Multilevel Graph Coarsening for ILU Preconditioning

Disclaimer

Matrix Reordering Using Multilevel Graph Coarsening for ILU Preconditioning

Daniel Osei-Kuffuor * Ruipeng Li † Yousef Saad†

November 12, 2014

Abstract

Incomplete LU factorization (ILU) techniques are a well-known class of preconditioners, often used in conjunction with Krylov accelerators for the iterative solution of linear systems of equations. However, for certain problems, ILU factorizations can yield factors that are unstable, and in some cases quite dense. Reordering techniques based on permuting the matrix prior to performing the factorization have been shown to improve the quality of the factorization, and the resulting preconditioner. In this paper, we examine the effect of reordering techniques based on multilevel graph coarsening ideas on one-level ILU factorizations, such as the level-based ILU(k) or the dual threshold ILUT algorithms. We consider an aggregation-based coarsening idea that implements two main coarsening frameworks - a top-down approach, and a bottom-up approach - each utilizing one of two different strategies to select the next-level coarse graph. Numerical results are presented to support our findings.

1 Introduction

Iterative methods based on Incomplete LU (ILU) preconditioners can be quite effective for solving certain types of linear systems of equations. Early work on the use of incomplete factorizations for preconditioning [3, 22, 32, 55, 75] led to the development of the level-based ILU(k) (or incomplete Cholesky, IC(k), for the symmetric case), see, e.g., [50,64,76]. Here, the underlying concept was founded on properties of M-matrices, and this generally resulted in poorer performance of ILU(k) preconditioners for general sparse systems. More robust alternatives, such as the threshold-based ILUT factorization, were later developed for general sparse matrices [51, 62].

However, for poorly structured matrices, ILU factorizations could generate significant fill-in, making the resulting L and U factors dense and hence inefficient as preconditioners. Reordering techniques borrowed from direct solution methods, such as the Reverse Cuthill-McKee (RCM) ordering [36], the minimum degree (MD) ordering [2, 37, 59, 70], and the nested dissection ordering [30, 35], have been used to help alleviate this problem, see for instance [8, 19, 29]. The paper [8], advocates the use of RCM along with ILU(1) and ILUT, especially for matrices that are far from being symmetric and diagonally dominant. Bridson et al. [19] provide an analysis and explain in particular why the RCM ordering can perform well while the MD ordering usually performs poorly for IC(0). Standard ILU factorizations are also prone to instability. First, there is the occurrence of very small pivots during the factorization, which can lead to a poor approximation to the original matrix. Column (or row) pivoting techniques have been typically employed to alleviate this [7, 31, 56, 64]. Some techniques permute the columns (or rows) of the sparse matrix during the factorization, so that columns (or rows) that are likely to yield unstable pivots are moved to the end.

A different type of instability is related to the fact that the L and U factors are themselves unstable, i.e., solving linear systems with them, will lead to unstable recurrences that grow exponentially. Such cases are

*Center for Applied Scientific Computing, L-561, Lawrence Livermore National Laboratory, Livermore, CA 94551. email: oseikuffuor1@llnl.gov. Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344.

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455. email: {rli,saad}@cs.umn.edu. Work supported by NSF under grant NSF/DMS-1216366 and by the Minnesota Supercomputer Institute

characterized by very large norms $\|L^{-1}\|$ and $\|U^{-1}\|$, see, e.g., [21,33]. Several techniques, such as modified ILU methods, see for instance [23, 28, 38, 45, 53, 72], shifted ILU methods, see for instance [48, 49, 57], as well as rigorous dropping strategies for ILUT, see [9], have been proposed in the literature to address this problem.

In recent years, there has been some interest in combining ILU factorizations with multilevel methods to obtain more efficient preconditioners for general sparse linear systems. One such example is the use of ILU(k), in particular ILU(0), as a smoother for the multigrid method [71]. In [9] and [10], Bollhöfer and co-workers used the idea of column pivoting, in combination with rigorous dropping strategies, to construct a multilevel preconditioner based on the Crout version of ILUT. One of the main ideas of multilevel methods is to define an ordering of the nodes in the adjacency graph of the matrix, by splitting the nodes into coarse and fine sets. This so-called C/F splitting is typically performed to promote independence among the nodes in the fine set. This allows for an efficient formation of an accurate approximation to the Schur complement matrix for the next level of the multilevel scheme. Furthermore, the nodes in the independent set can be eliminated simultaneously, which is beneficial for parallel computing. Independent set orderings [63], and generalized block-independent set orderings [66] have been exploited to develop multi-elimination ILU preconditioners with the goal of promoting parallelism. In [11], Ploeg, Botta and Wubs use a grid dependent reordering scheme to construct an effective multilevel ILU method for elliptic PDEs. Later on in [12], Botta and Wubs generalize this further and define an ordering that selects a nearly independent set of nodes that are diagonally dominant, and from which an independent fine set is extracted. This helps to promote stability in the factorization, while reducing the cost in the formation of the approximate Schur complement matrix for the next level. In [26], Chow and Vassilevski use the strength-of-connection idea from multigrid to define the C/F splitting of the nodes. Unlike some other procedures based on attaining a diagonally dominant fine set, this approach tries to ensure that the coarse set provides a good interpolation for the original problem. Non-symmetric C/F nodes partitioning approaches based on diagonal dominance ratio and sparsity combine multilevel graph coarsening strategies with standard threshold-based ILU factorizations to yield robust multilevel preconditioners for general sparse linear systems, see for example [46, 47, 65].

The goal of this paper is not to propose a new type of (multilevel) ILU techniques. Instead, the focus is to combine algebraic strategies with graph algorithms as a reordering tool to improve the stability of factored or ILU-type preconditioners. Thus, the work described in this paper may be more along the lines of earlier work presented in [18, 27], where ideas from graph theory are used to develop reordering techniques for ILU and factored approximate inverse preconditioners. In [18], a minimum inverse penalty (MIP) algorithm is developed by exploiting strategies to minimize the cost of the factorization and improve the accuracy of the factorization. The authors also present a weighted nested dissection strategy, and show its effectiveness on anisotropic problems. In [27] a minimum discarded fill (MDF) algorithm is presented as a reordering technique for ILU. The results show superior performance over traditional reordering strategies such as RCM, for unstructured grid problems. However, the cost of this reordering is generally higher than that of traditional methods. For both of these papers, the underlying observation is that for problems with anisotropy and inhomogeneity, reordering strategies that exploit the algebraic properties of the coefficient matrix are generally more effective than traditional methods.

In this paper, multilevel graph coarsening strategies are exploited to define an algebraic reordering technique for ILU. This constitutes one of the main differences between the work presented in this paper and previous work. It is worth noting that in [52, 54] the authors also make use of a reordering based on a multilevel graph coarsening strategy to develop an aggregation-based algebraic multigrid technique. However, the algebraic strategy for defining the coarsening is different. In this paper, we present two different strategies for determining which nodes are good candidates for the coarse or fine sets. We apply this reordering to the matrix prior to building its ILUT factorization. A multilevel dropping strategy is adopted, to be consistent with the level structure of the reordered system, and we show that the strategies described in this paper are also feasible with the level-based ILU factorization. In more formal terms, this paper seeks to answer the following question: *Is it possible to use ideas from multilevel graph coarsening to improve the quality of the ILU factorization?* In other words, we wish to incorporate multilevel coarsening ideas within a (single level) ILU framework to derive robust and effective preconditioners.

The paper is organized as follows: In Section 2, we briefly introduce the graph coarsening idea and discuss the multilevel framework. Section 3 presents the different strategies for performing the C/F splitting for the multilevel algorithm, and the full multilevel graph coarsening algorithm is presented in Section 4. In

Section 5, we describe the final reordering and discuss its adaptation to the ILU factorization. Section 6 gives a description of the model problems used in this paper. Numerical results are presented in Section 7, and we conclude in Section 8.

2 Graph Coarsening Strategies

Multilevel methods, such as multigrid [39, 60, 71] or Schur-based multilevel techniques [4, 6, 25, 66], rely on graph coarsening strategies to construct the next level matrix in the multilevel process. For multigrid, this corresponds to selecting a subset of the original or fine grid, known as the 'coarse grid', for which the solution to the residual equation $Ae = r$, for some error e and residual r, is slow to converge, see for instance [20]. For multilevel Schur-based methods, such as multilevel ILU, the coarsening strategy may correspond to selecting from the adjacency graph of the original matrix, a subset of nodes that form an independent set [66], or a subset of nodes that satisfy good diagonal dominance properties [65], or that limit growth in the inverse LU factors of the ILU factorization [9, 10].

The general idea of coarsening is as follows. Given a graph with n vertices, we would like to find a smaller graph, which yields a good representation of the original graph. Suppose we have an adjacency graph $G = (V, E)$ of some matrix A. For simplicity, we assume that G is undirected. An edge, e_{ij} represents the binary relation $(i, j) : a_{ij} \neq 0$. Coarsening the graph consists of finding a 'coarse' approximation $(\hat{V}, \hat{E})$ so that $|\hat{V}| < |V|$ and such that the new graph is a faithful representation of the original graph in some sense. By recursively coarsening the original graph, we obtain a hierarchy of approximations to the original graph. There are several techniques available for coarsening a graph. The work in this paper utilizes an aggregation-based approach, which is described next.

2.1 Pairwise Aggregation

Aggregation-based techniques are a common strategy for coarsening, see for instance [25, 52, 54, 71, 73, 74]. The pairwise aggregation strategy seeks to simply coalesce two adjacent nodes in a graph into a single node, based on some measure of nearness. The technique is based on edge collapsing [40], which is a well known method in the multilevel graph partitioning literature. In this method, the collapsing edges are usually selected using the *maximal matching* method. A *matching* of a graph $G = (V, E)$ is a set of edges $\tilde{E}$, $\tilde{E} \subseteq E$, such that no two edges in $\tilde{E}$ have a vertex in common. A maximal matching is a matching that cannot be augmented by additional edges to obtain another matching. There are a number of ways to find a maximal matching for coarsening a graph. We use a simple greedy strategy similar to the *heavy-edge matching* approach proposed in [41].

In algebraic terms, this greedy matching algorithm simply matches a node i, with its largest (off-diagonal) neighbor j_{max}, i.e., we have $|a_{ij_{max}}| = \max_{j \in adj(i), j \neq i} |a_{ij}|$, where $adj(i)$ denotes the adjacency (or nearest-neighbor) set of node i. When selecting the largest neighboring entry, ties are broken arbitrarily. If j_{max} is already matched with some node $k \neq i$ then node i is left unmatched and considered as a singleton. Otherwise, we match i with j_{max}. Once the matching is done, each pair of matched nodes are coalesced into a new coarse node. When two nodes i, and j_{max} are coalesced into a new one, the new node is considered the parent of the two nodes i and j_{max}, and denoted by $par(i, j_{max})$. As in standard agglomeration techniques, the parent node is represented by one of the nodes that combine to form the parent. That is, $par(i, j_{max}) = i$ or j_{max}. Singletons represent themselves as parents. Figure 1 gives an illustration of a single coarsening step. Nodes i and j_{max} are coalesced into node $par(i, j_{max})$ which will represent both nodes in the coarse-level graph. Moreover, a singleton node will be simply mapped to itself.

2.2 Multilevel Coarsening Scheme

The coarsening strategy can be expanded into a multilevel framework by repeating the process described above on the graph associated with the nodes in the coarse set. Let $G_\ell = (V_\ell, E_\ell)$ and define G_0 to be the original graph G and $G_1, G_2, \ldots, G_m$ be a sequence of coarse graphs such that G_ℓ is obtained by coarsening on $G_{\ell-1}$ for $1 \leq \ell \leq m$. Let $A_0 = A$ and A_ℓ be the adjacency matrix associated with graph G_ℓ. As previously mentioned, G_ℓ admits a splitting into coarse nodes, C_ℓ, and fine nodes, F_ℓ, so that the matrix A_ℓ can be reordered in one of the following ways:

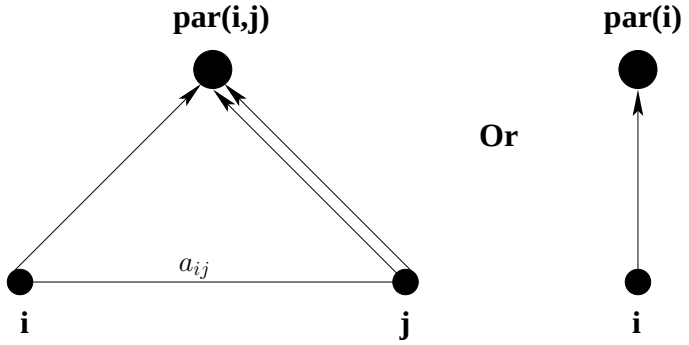


Figure 1: The coarsening process. Left: i and j_{max} are coalesced into node $par(i, j_{max})$ and the double-arrow shows that $par(i, j_{max})$ is represented by j_{max}; Right: singleton i is mapped into itself.

$$\begin{bmatrix} A_{C_\ell C_\ell} & A_{C_\ell F_\ell} \\ A_{F_\ell C_\ell} & A_{F_\ell F_\ell} \end{bmatrix} \tag{1}$$

or

$$\begin{bmatrix} A_{F_\ell F_\ell} & A_{F_\ell C_\ell} \\ A_{C_\ell F_\ell} & A_{C_\ell C_\ell} \end{bmatrix}. \tag{2}$$

We refer to the first approach as the *Bottom-Up approach*, since the coarse set, which is used to construct the next-level matrix, is put in the upper-left block. In this approach, the coarse set represents nodes that are considered "favorable" for the ILU factorization. The second approach is referred to as the *Top-Down approach* in which the next-level nodes are put in the lower-right block. Here, the coarse set represents nodes that are considered "unfavorable" for the ILU factorization. The criteria for determining which nodes are "favorable" and which are nodes are not is discussed later in Section 3. In both schemes, the coarser-level graph, $G_{\ell+1}$, and the corresponding adjacency matrix, $A_{\ell+1}$, are constructed from G_ℓ and A_ℓ. To construct $G_{\ell+1}$, we need to create edges, along with edge-weights, in certain ways between two coarse nodes created during the coarsening process. To do this, we first introduce some additional notation. Referring to the left side of Figure 1, if $t = par(i, j)$ is a node in G_{l+1} we will denote the representative node j in G_l by $r(t)$ (representative child) and the other node, i, by $c(t)$ (child). This defines two mappings $r(.)$ and $c(.)$ from G_{l+1} to G_l. In the figure we have $c(t) = i$ and $r(t) = j_{max}$ for the case on the left and $r(t) = i$ and $c(t) = i$ for the case on the right.

One way to define edges at the next level is to set two coarse nodes to be adjacent in $G_{\ell+1}$ if their representative children are adjacent in G_ℓ. With the notation just defined, this means that for any $x, t \in V_{l+1}$:

$$(t, x) \in E_{l+1} \quad \text{iff} \quad (r(t), r(x)) \in E_l.$$

This may be considered as a *one-sided* approach, since a fine node $c(t)$ in F_ℓ, does not contribute edges to the adjacency graph of its parent. The edge-weights for the new graph, $G_{\ell+1}$, using this approach, could be taken directly from the edges corresponding to the coarse nodes in the finer graph G_ℓ, i.e., entries of $A_{C_\ell C_\ell}$. In other words, we have $A_{\ell+1} = A_{C_\ell C_\ell}$. Although this way of creating edges in the new coarse graph is cheap and easy to implement, it could lead to coarse graphs that are poor approximations of the original graph as the coarsening scheme progresses. Since coarsening may also be seen as a way of 'breaking' connections in the fine level graph, it is easy to see how this one-sided approach could lead to a graph of singletons after a few coarsening steps. As a result, this approach is suitable only when a few levels of coarsening is sufficient for reordering the nodes.

An alternative to this one-sided approach is to define two coarse nodes to be adjacent in $G_{\ell+1}$ if they are parents of adjacent nodes in G_ℓ. With the notation introduced above this means that for any $x, t \in V_{l+1}$:

$$(t, x) \in E_{l+1} \quad \text{iff} \quad (r(t), c(t)) \times (r(x), c(x)) \cap E_l \neq \varnothing.$$

Let P_ℓ denote a corresponding interpolation operator for the above mapping. That is, the columns of P_ℓ contain nonzero weights in positions that correspond to nodes in the current-level graph that contribute

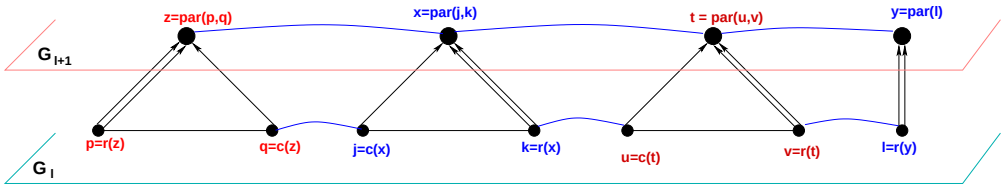


Figure 2: Two-sided approach for building the next-level coarsened graph

edges to the next-level coarse graph. Then $A_{\ell+1}$ is obtained from the Galerkin projection $A_{\ell+1} = P_\ell^T A_\ell P_\ell$. This way of defining edges in the coarse graph is quite common in multilevel graph partitioning techniques, see for instance [41], and other aggregation-based methods, see for instance [16,54,71]. It may be considered as a *two-sided* approach, since, unlike the previous approach, the fine nodes in F_ℓ do contribute edges to the adjacency graph of their parents. Figure 2 gives an illustration of the approach. The edge-weights of the new coarse graph are typically defined as the sum of the weights of the edges in the fine graph that connects their children, or some weighted average of this sum, see for instance [52,54,71].

3 Preselection Techniques

The coarsening strategy described in Section 2 relies on the strength of the connections between the nodes of the graph. Each node is compared to its largest off-diagonal neighbor (or most strongly connected neighbor) with respect to some defined weights. Which candidate nodes are labeled 'coarse' and which ones are labeled 'fine' during the coarsening process depends on their weights, which are saved in the vector $\mathbf{w}$. These weights are chosen to capture some intrinsic features of the matrix, or its adjacency graph, at the current level. In this paper, we focus on diagonal dominance as a criterion for defining the weights in $\mathbf{w}$, since we wish to apply some form of incomplete factorization on the resulting reordered matrix. The goal is to obtain a reordering of the matrix such that rows (or columns) corresponding to nodes that satisfy some diagonal dominance criterion are ordered first (in the top-left corner of Equations (1) and (2)). Next, we present two different strategies to define $\mathbf{w}$ of this type.

3.1 Strong Neighbor Connection Ratio

In this approach, weights are defined based on the relative size of the diagonal entry and its corresponding largest off-diagonal neighbor. We define the term γ_i as:

$$\gamma_i = \frac{|a_{ij_{max}}|}{|a_{ii}| + |a_{ij_{max}}|} \quad \text{with} \quad |a_{ij_{max}}| = \max_{j \in adj(i), j \neq i} |a_{ij}|.$$

Recall that the matched pair (i, j_{max}) is typically coalesced together during the coarsening process. One can think of γ_i as a measure of the degree of dependence of node i on node j_{max}. If γ_i is large, then node i strongly depends on node j_{max}. On the other hand, if γ_i is small, then the dependence of node i on node j_{max} is only a weak one. This notion of dependence is similar to that of 'strength of connection', which is commonly used when defining coarse grid operators in algebraic multigrid, see for instance [14,20,25,60,71]. The weights, w_i, are then formally defined as:

$$w_i = \gamma_i \times (1 + |adj(i)|)$$

The quantity $(1 + |adj(i)|)$ represents the number of non-zero entries in row i of the adjacency matrix (including the diagonal entry). It is used here to scale γ_i so that nodes with fewer non-zeros have a smaller entry in $\mathbf{w}$, and hence are more likely to be ordered first in the reordered system. This serves the goal of exploiting structure in the reordering to reduce fill-ins during the factorization of the reordered system.

The decision as to which node from the matched pair, (i, j_{max}), goes into the coarse or fine set, depends on the relative size of their weights, w_i and $w_{j_{max}}$, and on the direction of coarsening. For the bottom-up approach (see Equation (1)), at level ℓ of the multilevel process, the node with a smaller weight can yield a more favorable pivot for the subsequent incomplete factorization. Hence it will be a good candidate for the

coarse set, C_ℓ. Conversely, for the top-down approach (see Equation (2)), the node with a larger weight will be a good candidate for the coarse set. If $w_i = w_{j_{max}}$, then either node is put in C_ℓ and the other in F_ℓ.

3.2 Relaxation

Another way to define the components of $\mathbf{w}$ for coarsening is to use relaxation sweeps in a way similar to multigrid [15]. Relaxations are a common ingredient used in multigrid methods to damp out oscillatory errors in the residual equation. More recently, a technique known as compatible relaxation [5,13,17,34,44] has been used to select candidates for the coarse grid in algebraic multigrid (AMG). In this approach, relaxations are done on the homogeneous equation $A_\ell e_\ell = 0$, where e_ℓ is initially chosen to be in the near null space of A_ℓ. In [34], after a few relaxation sweeps, nodes that are quick to converge are put in F_ℓ. The remaining nodes (that are slow to converge) are judged good candidates for the coarse set and put in some set U_ℓ. An independent subset of these candidates is then selected and put in C_ℓ, and the process is repeated on the remaining candidates, $U_\ell \setminus C_\ell$, until a C/F splitting of the nodes is obtained. Notice that nodes in the F_ℓ set represent rows that are 'good' in terms of diagonal dominance with respect to the associated adjacency matrix A_ℓ.

In using relaxations to form $\mathbf{w}$, we follow an approach inspired by the compatible relaxation technique described above. We define γ as the vector resulting from applying a few relaxation sweeps to solve the linear system $A\gamma = 0$. Following [44] and exploiting the structure of the matrix, we then define the components of $\mathbf{w}$ as:

$$w_i = \frac{|\gamma_i|}{\|\gamma\|_\infty} \times (1 + |adj(i)|).$$

The weight vector $\mathbf{w}$ now contains a measure of the relative convergence of all the nodes, with respect to the slowest converging node, which is then scaled by the number of non-zero entries in the row in order to reduce fill-ins. For the top-down approach, if w_i is small, then node i is put in F, otherwise node i is a good candidate for the coarse set. For the bottom-up approach, however, small weights represent good candidates for the coarse set.

Recall that relaxation schemes, such as Jacobi or Gauss-Seidel, are not guaranteed to converge if the matrix A is not strictly or irreducibly diagonally dominant [64]. Thus, they tend to be ineffective on numerically challenging problems, such as ones that are highly indefinite or ill-conditioned. For such problems, it is possible to transform the system matrix into one that can be handled by the relaxation scheme. In this paper, we transform the original matrix A into a new matrix L_A, that represents some graph Laplacian derived from A. Relaxation-based schemes have been successfully used on graph Laplacian matrices as a means of defining algebraic distances between two nodes in some network graph, see for instance [24,58]. We define the edge weights of the graph Laplacian for the matrix, A, to be based on the neighbor connection ratio as follows:

$$l_{ij} = -\frac{|a_{ij}|}{|a_{jj}| + |a_{ii}|}$$

Here, the $l_{i,j}$ represent the off-diagonal entries of the matrix L_A. This formulation preserves symmetry in the off-diagonal entries, and the diagonal entries of L_A are then defined as:

$$l_{ii} = \sum_{j=1}^{n} |l_{ij}|$$

Note here L_A will receive an empty row when a_{ii} is the only nonzero entry in row i of A from the above definition. However, in practice these rows are actually rejected during the preselection stage.

We now have a more general approach to obtain the entries in $\mathbf{w}$ via relaxations, for an arbitrary matrix. That is, instead of performing relaxations on the homogeneous equation $A\gamma = 0$, we perform them on the system $L_A\gamma = 0$. The initial iterate, γ^0, for the relaxation scheme is chosen so that odd nodes have value $+1$ and even nodes have value -1.

3.3 Candidate set selection

In general, all the nodes in the current graph can be considered candidates for coarsening. However, in practice, not all the nodes need to be considered. Nodes that have very small entries in $\mathbf{w}$ can be considered

good enough to produce stable pivots during the ILU factorization, and hence can be immediately put in the set constituting the top-left block of the reordered system (in both the top-down and bottom-up schemes). During the preselection phase of the coarsening scheme, a node i, is selected as a candidate and put in the undetermined set U_ℓ for coarsening if w_i is large, relative to some threshold $\beta\bar{\mathbf{w}}$; where $\bar{\mathbf{w}}$ is the mean weight of $\mathbf{w}$ and β is some scaling tolerance. Thus, for a particular problem, changing the size of β will determine how many nodes are used for coarsening to construct the next level coarse graph.

4 The multilevel coarsening algorithm

Next we give a complete algorithm for the multilevel coarsening strategy used to reorder the system before performing the ILU factorization. The algorithm will describe the bottom-up approach, as the top-down approach follows a similar framework. The algorithm only gives a high-level description of the coarsening scheme, in order to keep it simple and more general. In what follows, let $p(i)$ denote a node that is matched with node i.

Algorithm 4.1. *Multilevel Coarsening Algorithm*

1. *Do matching on* $G_\ell = (V_\ell, E_\ell)$ *to obtain pairs* $(i, p(i))$
2. *For* $\ell = 0 : nlev$, *do:*
3. *Compute* $\mathbf{w}$ *and form candidate set* U_ℓ, *for coarsening*
4. *Initialize* $C_\ell = V_\ell \setminus U_\ell$
5. *Initialize* $F_\ell = \emptyset$
6. *Obtain a* C/F *splitting of* U_ℓ *to get* I_C *and* I_F
7. *Set* $C_\ell = C_\ell \bigcup I_C$
8. *Set* $F_\ell = F_\ell \bigcup I_F$
9. *Build next level graph* $G_{\ell+1}$
10. *EndDo*

Algorithm 4.1 performs $nlev + 1$ levels of coarsening, starting with the original fine graph G_0. The algorithm begins by performing a matching on the current level graph G_ℓ to obtain the pairs $(i, p(i))$, which will be used to split the candidate set, U_ℓ. This matching promotes independence among the nodes in the coarse set. Furthermore, it also dictates the structure of the resulting reordered matrix, which in turn affects the fill-in pattern of the ILU factorization applied to the matrix. In line 3 of the algorithm, $\mathbf{w}$ is computed and a candidate set, U_ℓ, is extracted for coarsening. The nodes that do not belong to U_ℓ are assigned to the coarse set, C_ℓ (line 4), as they represent 'good' nodes that will be ordered first in the final reordering. A C/F splitting of nodes in U_ℓ is done to obtain coarse nodes, I_C, and fine nodes, I_F, which are used to update C_ℓ and F_ℓ (in lines 7 and 8). Finally, the next level graph is assembled using either the one-sided approach or the two-sided approach for defining the edge weights, discussed earlier in section 2.2.

The overall quality of the coarsening scheme is determined, in part, by the strategy used to obtain the C/F splitting in line 6. A common approach is to use a strategy that promotes independence among the nodes in the coarse set. Another approach, common in the multilevel ILU literature, focuses on promoting diagonal dominance among the nodes in the coarse set. The strategy followed in this paper uses the pairwise aggregation technique to obtain a splitting so that the nodes in the coarse set are good in terms of diagonal dominance, and are *nearly* independent. Algorithm 4.2 formally describes the technique used to obtain the C/F splitting in line 6 of Algorithm 4.1.

Algorithm 4.2. *Coarse step (C/F splitting) Algorithm*

1. *For* $i \in U_\ell$, *do:*
2. *Set* $mark(i) = 0, \forall i \in U_\ell$
3. *Set* $j = p(i)$
4. *If* $mark(i) == 0$ *and* $mark(j) == 0$ *and* $i \neq j$, *then*
5. *If* $w_i \leq w_j$, *then*
6. *Set* $i \in I_C$ *and* $j \in I_F$
7. *Else*
8. *Set* $i \in I_F$ *and* $j \in I_C$

9. *Set* $mark(i) = mark(j) = 1$
10. *EndIf*
11. *EndDo*
12. *Assign any remaining nodes to* I_C

Initially, all the nodes in U_ℓ are unmarked. A pair of unmarked nodes, i and j, that are matched, are compared to each other based on the relative size of their entries in $\mathbf{w}$, and assigned to the coarse set, I_C, or the fine set, I_F. These nodes are then marked, and the process is repeated until all matched nodes in U_ℓ, that are unmarked, have been marked and assigned to I_C or I_F. Finally, to complete the splitting, any remaining nodes that are not marked are assigned to the coarse or fine set. These nodes come from two different contributions. The first consists of nodes that are matched to themselves in the original graph (i.e. singletons). These nodes represent diagonal entries in the associated matrix, and hence will be good candidates for the coarse set. The other contribution comes from nodes that are matched with neighbors that have been previously marked. Note that the matching described in section 2.1 is not one-to-one. Hence a node, whose matched neighbor is involved in another matching, could be skipped by the check in line 4 of Algorithm 4.2. These nodes, which now behave like singletons, can be simply assigned to the coarse set to be dealt with at the next level of coarsening. Alternatively, they may be assigned to either I_C or I_F depending on the size of their entry in $\mathbf{w}$, relative to that of their matched neighbor (albeit this neighbor is already marked). In either case, the notion of independence is compromised, as the unmarked nodes may be assigned to the same set as their matched neighbor. In the former case, independence could be recaptured at the next level as the matched pair remain neighbors at the next level. In the latter case, we obtain a C/F splitting that adheres more strictly to the requirement that 'good' nodes be assigned to the coarse set. The result is a trade-off between the diagonal dominance and the independence of the nodes in the coarse set. In our numerical experiments, we follow the latter case.

5 Final reordering and ILU factorization

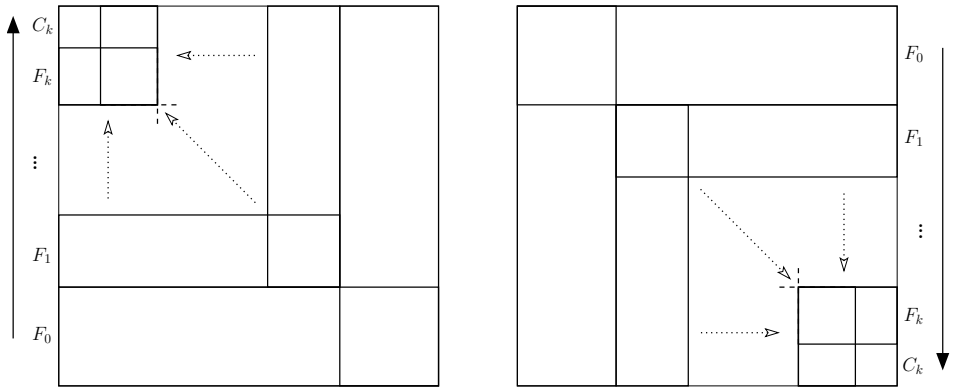
5.1 Final reordering

Using Algorithm 4.1, after each step of coarsening we obtain a C/F splitting of the current graph G_ℓ, into the sets C_ℓ and F_ℓ. For the bottom-up approach described above, after k steps of coarsening, we obtain the following sequence of sets: $C_k, F_k, \ldots, F_0$. This sequence represents the final reordering of the system matrix. Nodes in C_k will be ordered first, since they represent the 'best' nodes selected by the coarsening strategy. This will be followed by nodes in F_k, then F_{k-1} and so on, until F_0, which contains the nodes deemed to be 'worst' by the coarsening strategy. Considering each of these sets as a single level, we obtain $k+1$ levels in the final reordered matrix.

For the top-down approach, the order of the sequence is reversed, since coarsening is done on the 'worst' nodes. Thus, nodes in F_0 represent the 'best' nodes and will be ordered first, and nodes in C_k will be ordered last. Figure 3 shows an illustration of the final reordering for the different coarsening approaches.

5.2 Adaptation to ILU factorization

Once the matrix has been reordered, an incomplete LU factorization is performed. In this paper, we consider the column-based implementation of two ILU methods: the incomplete LU factorization with level of fill dropping (ILU(k)); and the incomplete LU factorization with dual threshold dropping (ILUT) [64]. The ILU factorization is an approximation to the Gaussian elimination process, where small non-zero entries generated during the factorization, are dropped by some dropping criterion. During the factorization process, new non-zero entries known as 'fill-ins' may be generated at locations occupied by zero elements in the original matrix. For the ILU(k) factorization, a 'level-of-fill' parameter, k, is used to control the amount of fill-ins allowed in the L and U factors [76]. For $k = 0$, the ILU factorization drops all fill-in elements generated during the factorization process. The factors, L and U, generated as a result, will have the same pattern as the lower and upper triangular parts of A respectively. The level-of-fill concept associates each fill-in element with the 'level' at which it was formed. Initially, all non-zero entries of the matrix are assigned level zero. Then level k fill-ins are generated by products of fill-in elements at levels less than k. The ILU(k) factorization results



(a) Bottom-up approach (b) Top-down approach

Figure 3: Final reordering of the original matrix after k steps of coarsening using the bottom-up approach (left) and top-down approach (right).

from keeping all fill-ins that have level k or less, and dropping any fill-in whose level is higher. For the ILUT factorization, the strategy for dropping small terms is based on two parameters [62]. The first parameter is a drop tolerance, τ, which is used mainly to avoid performing an elimination if the pivot is too small. The second parameter is an integer, ρ, which controls the number of entries that are kept in the j-th columns of L and U. Further details of the column-based implementation of the ILUT factorization can be found in [45].

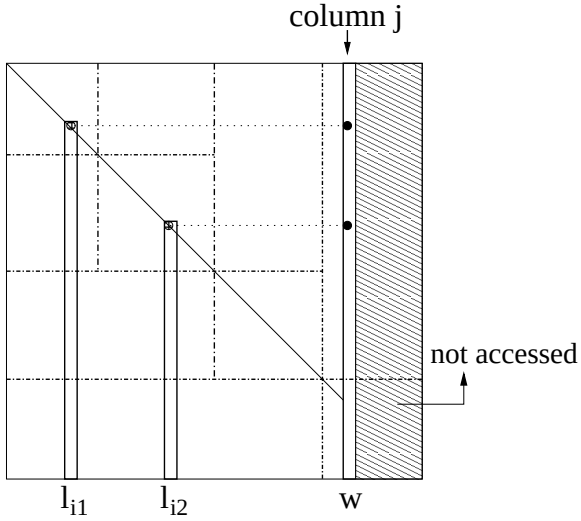


Figure 4: Column-based ILUT on reordered matrix with four levels of coarsening (bottom-up scheme).

The standard ILUT algorithm can be modified to take advantage of the reordering derived from the coarsening strategies discussed above. One simple approach is to use dropping strategies that are local to each level of the reordered matrix. In other words, if the coarsening strategy yields $\ell + 1$ levels for the reordered matrix, then the drop tolerance, τ, and the fill-level parameter, ρ, will each take the form of a vector of length $\ell + 1$, so that $\tau(0)$ and $\rho(0)$ dictate the dropping rules for the factorization of the level-zero block of columns, and so on. An illustration of the elimination process is shown in Figure 4, and a sketch of the general structure of the algorithm is given next as Algorithm 5.1. Here, we assume that **lev** is a vector of length $\ell + 1$, containing the block size (number of columns) for each level.

Algorithm 5.1. *Left-looking or JKI ordered ILUT*

0. Set $bsize = \mathbf{lev}(0)$
1. Set $\ell = 0$
2. For $j = 1 : n$, *do:*
3. $\mathbf{z} = A_{1:n,j}$
4. If $j > bsize$
5. $\ell = \ell + 1$
6. $bsize = bsize + \mathbf{lev}(\ell)$
7. EndIf
8. For $k = 1 : j - 1$ *and if* $z_k \neq 0$, *do:*
9. Apply first dropping rule to z_k *using* $\tau(\ell)$
10. If z_k *is not dropped,* $\mathbf{z}_{k+1:n} = \mathbf{z}_{k+1:n} - z_k \cdot L_{k+1:n,k}$
11. Enddo
12. For $i = j + 1, \ldots, n$, $l_{i,j} = z_i/z_j$ $(l_{j,j} = 1)$
13. Apply second dropping rule to $L_{j+1:n,j}$ *using* $\tau(\ell)$ *and* $\rho(\ell)$
14. For $i = 1, \ldots, j$, $u_{i,j} = z_i$
15. Apply second dropping rule to $U_{1:j-1,j}$ *using* $\tau(\ell)$ *and* $\rho(\ell)$
16. Enddo

In the following discussion, $l_{i,k}$ and $u_{i,k}$ represent the scalar entries at the i-th row and k-th column of the matrices L and U, respectively, $A_{1:n,j}$ denotes the j-th column of A, $\mathbf{z}_{k+1:n}$ denotes the last $n-k$ entries in the vector $\mathbf{z}$, $L_{k+1:n,k}$ denotes the last $n-k$ entries in the k-th column of L, $U_{1:j-1,j}$ denotes the first $j-1$ entries in the j-th column of U, and so forth. The algorithm begins by setting the block size parameter, $bsize$, to the size of the first level block. This block contains the columns (or rows) associated with the nodes adjudged to be the 'best' according to the coarsening strategy. At the start of a given step j, a working column, $\mathbf{z}$, is created and set to the initial j-th column of A, $\mathbf{a}_j$, on which elimination operations will be performed. The index j is then compared to $bsize$ to determine which parameters are to be used for the elimination. The level counter ℓ, and $bsize$ are then updated accordingly (lines 5 and 6). At the beginning of the elimination process (line 9), the pivot, z_k, is dropped if it is smaller relative to some scaling parameter based on $\tau(\ell)$. Otherwise, operations of the form $\mathbf{z} := \mathbf{z} - z_k\mathbf{l}_k$ are performed to eliminate entries of $\mathbf{z}$ from top to bottom, until all entries strictly above the diagonal are zeroed out. Here, $\mathbf{l}_k$ is used to denote the k-th column of L. The pivot entries, z_k, that are not dropped by the first dropping rule, will become the entries in the j-th column of U (line 14), while the entries below the diagonal of the updated vector $\mathbf{z}$, will be scaled by the diagonal entry, and become the j-th column of L (line 12). In Lines 13 and 15, the threshold, $\tau(\ell)$, is invoked again to drop small terms, then the largest $\rho(\ell)$ entries in the resulting i-th columns of L and U are kept, and the others are dropped.

By adopting a level-based dropping strategy, we have better control of how to drop small terms during the factorization at the different levels. Like standard ILUT, from a practical standpoint, it is often better to keep $\rho(\ell)$ large and vary $\tau(\ell)$ to control the dropping. One simple approach is to choose $\tau(0)$ to be relatively large for the first block, and (progressively) smaller for the subsequent blocks during the factorization. This is justified since the first block (or first set of blocks) contains nodes that are considered to have good diagonal dominance. Thus dropping during the factorization of the columns associated with these nodes could be done in a more aggressive manner. At later levels, the columns are considered to be 'poor' in terms of diagonal dominance, and a more accurate factorization may be needed to yield accurate and stable factors.

A similar approach may be used for the level-based ILU(k) method. During the factorization, the level-of-fill parameter k, can be (progressively) increased at each subsequent level from the first level. Like the ILUT method, this offers better control of how terms are dropped during the factorization.

Figure 5 shows the nonzero pattern of the 2D Laplacian matrix originating from the finite difference discretization of the Laplace operator on a 50×50 rectangular grid, with homogeneous Dirichlet boundary conditions. To illustrate the effect of the reordering strategy, Figure 6 shows the nonzero pattern of the same matrix reordered by the top-down version of the multilevel graph coarsening scheme. The figure depicts an example of the pattern of the reordered matrix, and the corresponding pattern of $(L+U)$ from the resulting ILUT factorization, after 3, 4, and 5 levels of coarsening. For each of the reordered matrices, the number of non-zeros in $(L+U)$ is approximately 2.5 times the number of non-zeros of the original matrix.

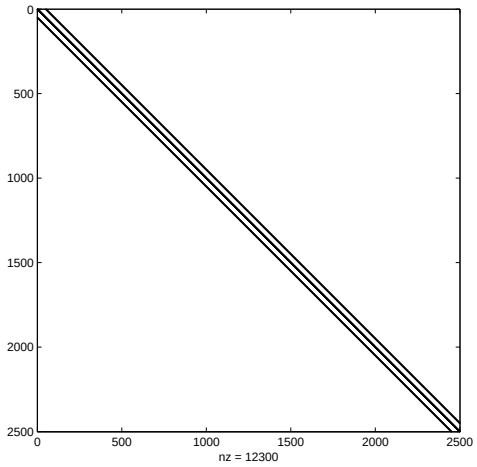


Figure 5: Nonzero pattern of the 2D Laplacian matrix.

5.3 Local reordering within the blocks

Given the level structure of the multilevel graph coarsening approach, a natural extension is the possibility of imposing some reordering within the block structure. For example, one could utilize a fill reducing technique such as RCM within each or some of the blocks in the multilevel hierarchy. This can help to further reduce fill-ins introduced during the ILU factorization, particularly when a high fill level is required for convergence. A typical scenario is when the candidate set selection phase (line 2 of Algorithm 4.1) selects only a small subset of nodes on which the coarsening is done. The resulting top-left block of the reordered matrix can be quite large and ordered in a way that is similar to that of the original matrix. Thus, reordering the nodes corresponding to this block by a fill reducing strategy like RCM can lead to a reduction in the amount of fill-ins that are discarded, and potentially yield a more accurate ILU factorization. Note that in general, at each level, the coarsening strategy produces a C/F splitting of roughly equal sizes. However, when the next coarse level graph is relatively small, then performing a local reordering is no longer necessary or beneficial. Hence it is sensible to utilize local reordering within each level when only a few levels of coarsening are performed, i.e. when $nlev$ in Algorithm 4.1 is small.

Incorporating local reordering requires only a minor modification to the multilevel graph coarsening algorithm. We insert the line

7a. Reorder F_ℓ

after Line 7 of Algorithm 4.1 to reorder the fine set at each level and the line:

9a. Reorder final coarse set C_{nlev}

after Line 9 of Algorithm 4.1 to reorder the final coarse set. We refer to this algorithm as the Multilevel Coarsening Algorithm with Local Reordering or 'Algorithm 4.1 with reordering.'

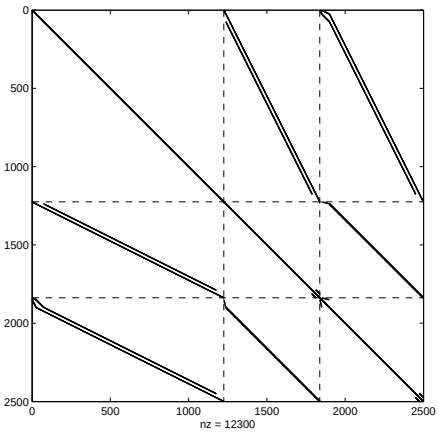
6 Test Problems

We present some numerical results using the new reordering strategy on various application problems. These problems are either indefinite, or exhibit numerical anisotropy that is derived from either the discretization grid or the equation coefficients. Details of the model problems are given next.

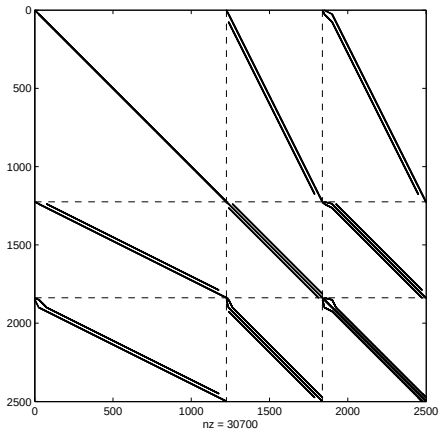
6.1 Problem I: 2D Helmholtz equation application

The Helmholtz equation is a partial differential equation of the form

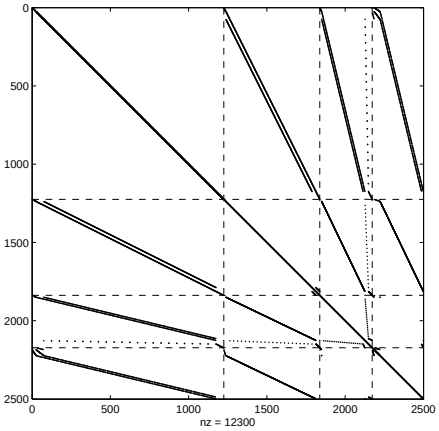
$$(\Delta + \omega^2)u = f, \tag{3}$$



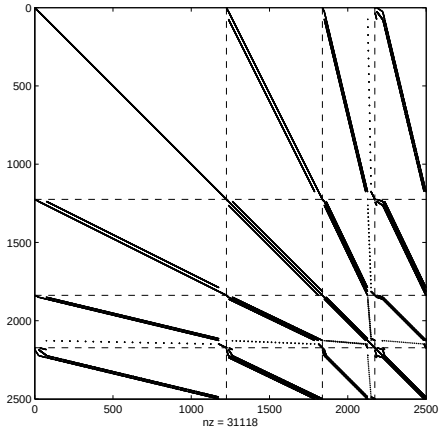
(a) Two levels of coarsening



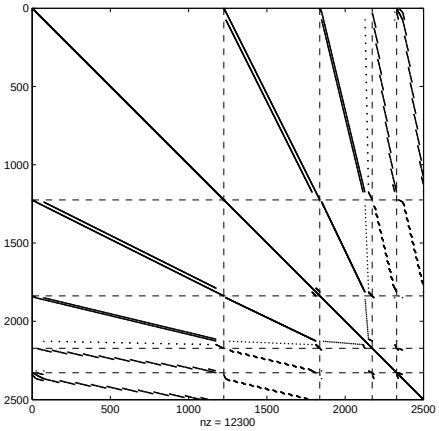
(b) Pattern of $(L+U)$



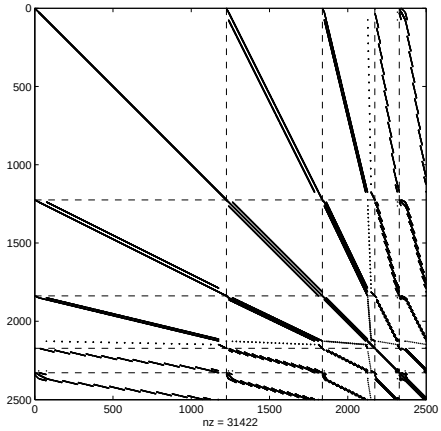
(c) Three levels of coarsening



(d) Pattern of $(L+U)$



(e) Four levels of coarsening



(f) Pattern of $(L+U)$

Figure 6: An example of the nonzero pattern of the reordered matrix and the resulting ILUT factorization for the 2D Laplacian.

which describes the propagation of waves in media. In the above equation, f represents a harmonic source, and ω represents the wavenumber. The numerical solution to the Helmholtz equation at high wave numbers has been the subject of extensive research. At high wave numbers, the system matrix tends to be very indefinite, causing problems for many numerical methods. Our application problem is based on the simulation of the diffraction of an acoustic wave originating from infinity through an open medium, and incident on a bounded obstacle with a circular boundary. Here, we assume the plane wave is propagating along the x-axis, and the radius of the bounded obstacle is 0.5m. For a suitable numerical solution to the problem by the finite element method, an artificial boundary condition is imposed at a distance 1.5m from the obstacle, using the Dirichlet-to-Neumann technique to satisfy the Sommerfeld radiation condition [43]. The resulting boundary value problem is discretized by the Galerkin least-squares finite-element method, using an isoparametric discretization over quadrilateral elements. The discretized problem is complex symmetric indefinite, and has size $n = 57,960$, with $nnz = 516,600$ nonzero elements.

In our numerical results, we use the ILUT factorization to construct the preconditioner for the numerical solution of the Helmholtz problem. We solve the linear system for increasing values of the wavenumber ω. As the wavenumber increases, the problem becomes more indefinite, which makes it challenging to solve by standard ILUT. Furthermore, since we consider a fixed grid, increasing the wavenumber makes the overall mesh resolution (measured in number of points per wavelength, ppw) decreasing as we increase ω (from the highest of 80 ppw for $\omega = 4\pi$ to a low of 10 ppw for $\omega = 32\pi$), leading to more challenging problems. As shown in previous work, see for instance [43,45,57], standard ILUT applied to this problem fails to converge at high wavenumbers. In such cases, a shifted ILUT factorization may be more appropriate. In the following numerical examples, we use the shifted ILUT method based on the τ-based scheme (i.e. shifting based on the drop tolerance) described earlier in the literature [57].

6.2 Problem II: Stretched circle problem

The model problem is a triangular finite element discretization of a 2D isotropic diffusion problem, defined on the unit circle. We consider two examples of sizes $n = 13,373$ and $n = 53,069$, corresponding to the mesh resolutions $h = 0.02$ and $h = 0.01$ respectively. For each of these meshes, the $y-$coordinates of the vertices of the resulting triangulation is stretched by a factor of 100, yielding a symmetric positive definite coefficient matrix that has strong unstructured anisotropic behavior. See [69] for more details of the problem description.

6.3 Problem III: Unstructured triangles problem

We consider a finite element discretization of

$$0.01\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f. \tag{4}$$

The problem is initially discretized on a mesh obtained from the mesh generation package NETGEN [67,68]. This mesh contains 474 unstructured tetrahedral elements and 145 vertices, on the unit cube. As a result, the coefficient matrix exhibits a non-grid-aligned anisotropic behavior. For our numerical experiments, the initial mesh is uniformly refined 4 times, using the finite element package MFEM [1], so that the resulting coefficient matrix has size $n = 337,105$.

6.4 Problem IV: Rotated anisotropy problem

Finally, we consider a 2D rotated anisotropic diffusion problem described earlier in the literature, see Equation 1 of [69]. The problem satisfies a biquadratic finite element discretization of

$$-(c^2 + \epsilon s^2)\frac{\partial^2 u}{\partial x^2} - 2(1-\epsilon)cs\frac{\partial^2 u}{\partial x \partial y} - (\epsilon c^2 + s^2)\frac{\partial^2 u}{\partial y^2} = f, \tag{5}$$

where $\epsilon = 0.001$, and c and s represent the cosine and sine of the angle of rotation, θ, respectively. For this model problem, we consider two examples, each of size $n = 261,121$, and corresponding to $\theta = \pi/16$ and $\theta = 2\pi/16$.

7 Numerical Results

The experiments were conducted sequentially on a single core of a 64-bit Linux cluster, with sixteen 2.60GHz Intel Xeon processor cores per node, and 20MB cache and 32GB memory per node. The code was compiled by the gcc compiler using -O3 optimization level. In the following experimental results, we test the effect of reordering by the multilevel graph coarsening strategy (MLGC) described above, and compare its performance against those of the natural (or original) ordering obtained from the discretization, as well as standard reordering techniques for ILU. In particular, we compare results against those of the reverse Cuthill-McKee algorithm (RCM), the multiple minimum degree algorithm (MMD), and the nested dissection algorithm (ND). We use the version of these algorithms as implemented in METIS [42].

The numerical solution for each example uses a right-preconditioned restarted GMRES [61] accelerator, with a restart dimension of 100. We note that this choice of the Krylov subspace dimension is rather liberal and a more conservative choice may be used in many of the test problems presented, leading to a more memory efficient solution strategy. However, we found that for some examples, using a smaller Krylov subspace dimension fails to produce converged results within the allowed number of iterations for some of the reordering schemes used in our comparisons with the MLGC scheme. Therefore, we fix the restart dimension to 100 in order to benefit presentation of sufficient comparative results. The maximum number of GMRES iterations is fixed at 300, and unless otherwise stated, we assume convergence when the ℓ_2-norm of the initial residual is reduced by a relative factor of 10^8. The right-hand side vector is artificially generated by assuming a solution of all ones, and we use a zero initial guess for the iterative solver. As discussed at the end of Section 5.2, the parameters which control the droppings in the ILU factorizations can be set in a progressive manner over the levels of the coarsening. Specifically, for the ILUT method, we use $\tau(i) = 0.8\tau(i-1)$, for levels $i > 0$. For the ILU(k) method, unless otherwise specified, we set the level-of-fill parameter k at some level i of the multilevel heirarchy, to $k = k_o + 1$, where k_o is the value of the level-of-fill parameter at level $i-1$ of the multilevel structure. In practice, we find that it is ideal to set the tolerance used to define the threshold for the candidate set selection to be $O(\tau)$, where $\tau = \tau(0)$ is the initial drop tolerance parameter for ILUT. Therefore, we choose $\beta = \tau$ when the ILUT method is used, otherwise (for ILU(k)) β is chosen so that the resulting fill factor (or memory used) for the preconditioner is similar to that of RCM. Furthermore, to simplify the presentation of results, we use the one-sided approach to build the coarse level graph whenever only one level of coarsening is performed. Otherwise, the two-sided approach is used instead.

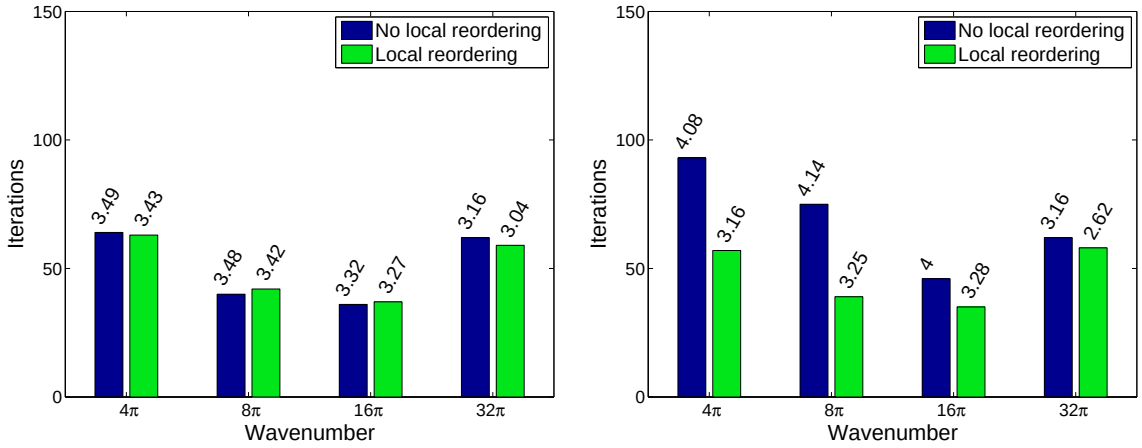
7.1 Comparison of MLGC variants

In this section, we compare the different variants of the MLGC algorithm on the different application problems using different options for the preconditioner.

7.1.1 Helmholtz application problem

We begin with a comparison of the performance of the MLGC algorithm on the Helmholtz problem with variable wavenumbers, using shifted ILUT as the preconditioner. For this example, we do two levels of coarsening for each test case and we vary τ (and β) for each value of the wavenumber. Each value of τ corresponding to a particular wavenumber, is the same across the different test cases. We use $\tau = 0.002$, 0.003, 0.006, and 0.02, respectively for the Helmholtz problem with wavenumbers $\omega = 4\pi$, 8π, 16π and 32π. In the results that follow, we use the 2-sided approach to build the coarse level graph.

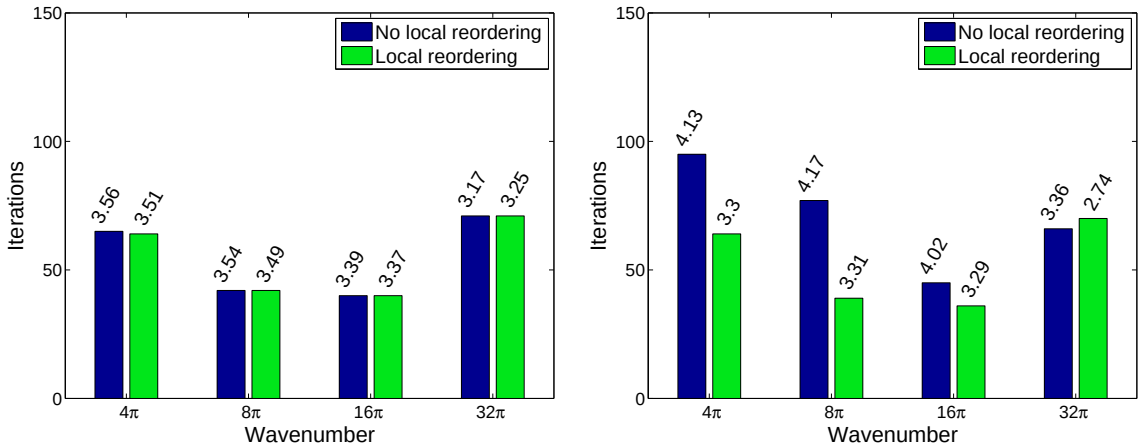
Figure 7 shows results for the algorithm using the strong neighbor connection ratio (MLGC-SNCR) approach for coarsening. Here, and in all other figures of this type, each bar represents a test problem; the height of each bar represents the number of iterations taken to solve the problem; and the number on top of each bar represents the corresponding fill factor for the preconditioner. This is defined as $(nnz(L + U))/nnz(A)$, which is the ratio of the number of nonzero elements needed to store the L and U factors over the original number of nonzero entries of A. Figure 7a shows the results using the top-down approach, with and without local reordering; and Figure 7b shows the corresponding results for the bottom-up approach. The results indicate that the top-down approach has only a marginal benefit in terms of memory cost and number of iterations when local reordering is performed. In contrast, the bottom-up approach shows more significant performance gains when local reordering is performed.



(a) top-down ordering

(b) bottom-up ordering

Figure 7: Shifted ILUT using the top-down and bottom-up variants of the MLGC-SNCR algorithm, with 2 levels of coarsening and $\beta = \tau = \{0.002, 0.003, 0.006, 0.02\}$, with and without local reordering, on the Helmholtz problem. The numbers at the top of each bar show the fill factors.



(a) top-down ordering

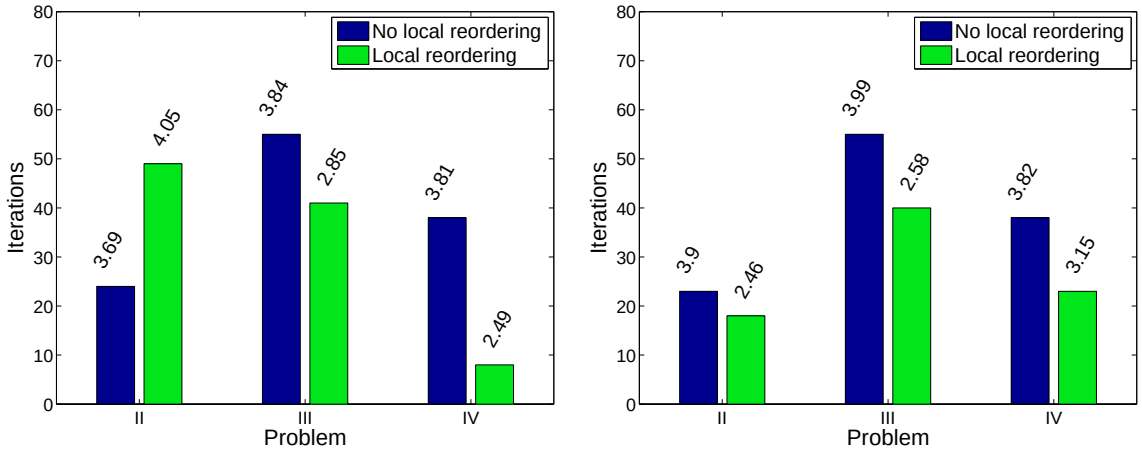
(b) bottom-up ordering

Figure 8: Shifted ILUT using the top-down and bottom-up variants of the MLGC-Relax algorithm, with 2 levels of coarsening and $\beta = \tau = \{0.002, 0.003, 0.006, 0.02\}$, with and without local reordering, on the Helmholtz problem. The numbers at the top of each bar show the fill factors.

Figure 8 shows results for the MLGC algorithm using relaxations for defining the coarsening strategy (MLGC-Relax). The observed results are quite similar to that of the MLGC-SNCR algorithm. The top-down approach shows almost no effect from local reordering, whereas the bottom-up approach benefits in terms of both performance and memory cost. The convergence performance of both the MLGC-SNCR and MLGC-Relax algorithms is similar. However, the MLGC-Relax approach appears to have a slightly higher memory cost for these examples.

7.1.2 Anisotropic examples

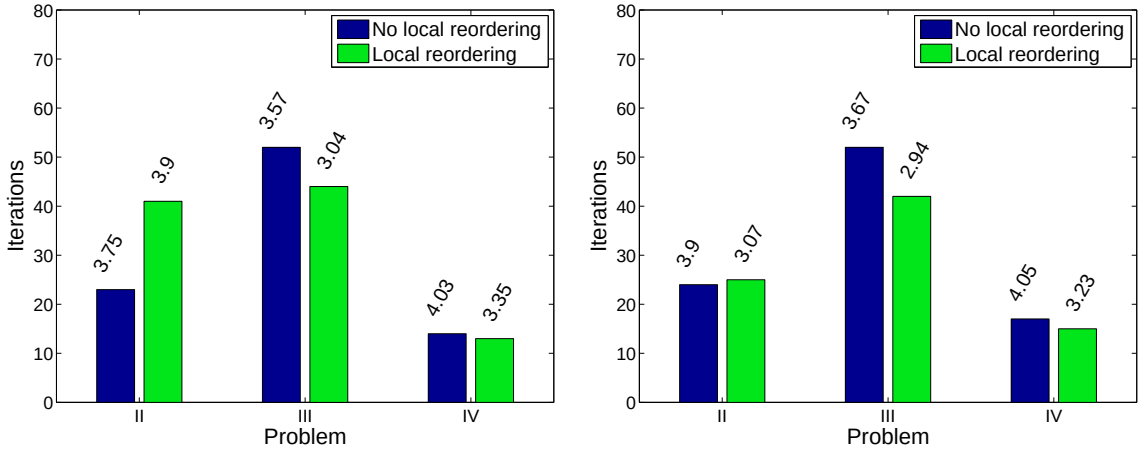
Next, we show the performance of the MLGC algorithm on some anisotropic examples using both ILUT and ILU(k) for preconditioning. The test problems considered are Problem II with a mesh size of 0.02, Problem III, and Problem IV with a rotation of $2\pi/16$.



(a) top-down ordering

(b) bottom-up ordering

Figure 9: ILUT using the top-down and bottom-up variants of the MLGC-SNCR algorithm, with 2 levels of coarsening and $\beta = \tau = \{0.005, 0.02, 0.005\}$, with and without local reordering, on the anisotropic problems.



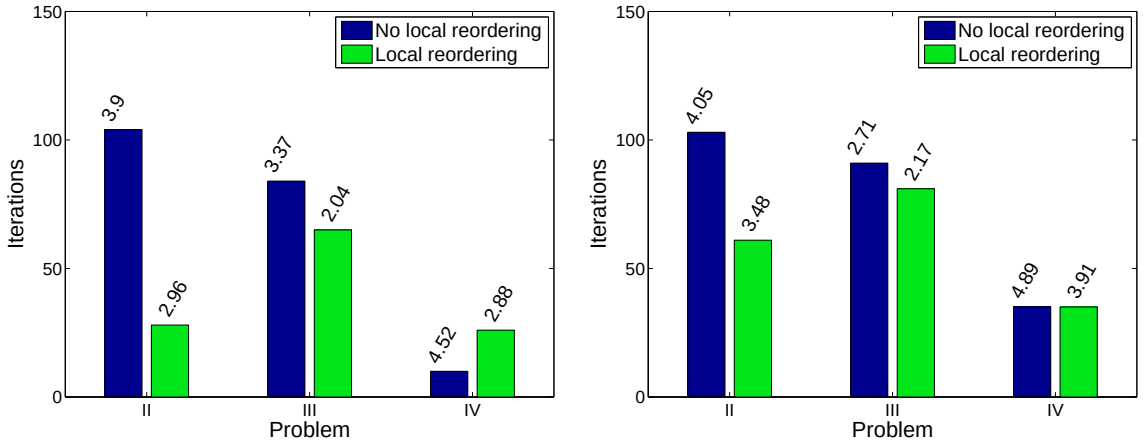
(a) top-down ordering

(b) bottom-up ordering

Figure 10: ILUT using the top-down and bottom-up variants of the MLGC-Relax algorithm, with 2 levels of coarsening and $\beta = \tau = \{0.005, 0.02, 0.005\}$, with and without local reordering, on the anisotropic problems.

For the tests involving ILUT, we do two steps of coarsening and for each problem, we choose τ to avoid large fill factors as follows: $\tau = 0.005$ for Problems II and IV, and $\tau = 0.02$ for Problem III. Figures 9 and 10 show the results for ILUT using the MLGC-SNCR and MLGC-Relax algorithms respectively. For the MLGC-SNCR algorithm, the results indicate that the bottom-up ordering scheme benefits from local reordering for all the test problems. However, the top-down approach only shows improved performance with local reodering for Problem III and Problem IV; Problem II appears to perform better without local reordering. Similar results are observed when the MLGC-Relax algorithm is used. The results also indicate that with the exception of a few cases, the MLGC-Relax algorithm generally has a higher memory cost compared to the MLGC-SNCR algorithm.

For the tests using ILU(k), we use a variable level-of-fill value that is chosen progressively over the levels of coarsening in the MLGC algorithm. In order to keep the resulting fill-factor similar to that attained for



(a) MLGC-SNCR (b) MLGC-Relax

Figure 11: ILU(k) with variable k, using the MLGC-SNCR and MLGC-Relax algorithms, with 1 level of coarsening and $\beta = \tau = \{0.005, 0.02, 0.005\}$, with and without local reordering, on the anisotropic problems. Problem II begins with ILU(3) and the remaining problems begin with ILU(1).

the tests with ILUT, we do one level of coarsening. For the test involving Problem II, we set the minimum fill level parameter for the ILU(k) factorization to be $k = 3$ for the first set of rows in the multilevel hierarchy, and $k = 4$ for the subsequent rows. For Problems III and IV, we set $k = 1$ for the first set of rows, and $k = 2$ for the subsequent rows. Since we perform only one level of coarsening, the bottom-up and top-down approaches yield identical results. Thus, we show only one result each from using the MLGC-SNCR algorithm and the MLGC-Relax algorithm. For consistency, we use the same values of β used for the cases with ILUT. Figure 11 shows the results. For both the MLGC-SNCR and MLGC-Relax algorithms, the results indicate significant gains in memory usage (over over $\approx 25\%$ for the MLGC-SNCR case) when local reordering is performed. The results also indicate better convergence in terms of iteration count for most of the test cases. The only exception is Problem IV, where the MLGC-SNCR algorithm converges in fewer iterations when no local reordering is performed, although at a much higher memory cost.

The above results show the effectiveness of the different variants of the MLGC algorithm on problems with indefinite coefficient matrices and anisotropic coefficient matrices. In general, the results indicate similar performance for both the MLGC-SNCR and the MLGC-Relax methods. However, the MLGC-Relax method typically yields a slightly more expensive preconditioner. The results also suggest that performing some local reordering in the multilevel hierarchy can help to improve the quality of the subsequent ILU factorization, and lead to a more effective preconditioner. We note, however, that how much a problem benefits from local reordering depends on several factors such as the type of local reodering utilized, the coarsening strategy used, and the size of the local blocks. In the next sections, we present some results comparing the performance of the MLGC algorithm to that of standard ordering techniques.

7.2 Helmholtz equation application

In the results that follow, we compare the performance of shifted ILUT with the natural ordering, and shifted ILUT with RCM, ND, MMD, and the MLGC orderings to solve the linear system originating from the Helmholtz application problem, for different wavenumbers. Here, we use the MLGC-SNCR algorithm since it gave similar results to the MLGC-Relax variant (see Figures 7 and 8). The MLGC strategy uses the top-down direction of coarsening, with 3 steps of coarsening, and the two-sided approach for computing the weights for the coarse level graph. To allow for a fair comparison, the drop tolerance for ILUT is adjusted so that the memory usage is kept at ≈ 3.5 for each preconditioner. As a result, β, used for the candidate set selection is set to $\beta = \tau = 0.002$, 0.003, 0.0055, and 0.016, respectively for the problem with wavenumbers $\omega = 4\pi$, 8π, 16π, and 32π. Table 1 shows the total number of iterations and the total time taken to construct the preconditioner and solve the linear system. The table also shows the error in the numerical solution, for

Ordering	k	No. iters	Fill factor	Total time	Error
Natural	4π	157	3.49	3.14	7.4×10^{-5}
	8π	170	3.47	3.39	4.4×10^{-5}
	16π	191	3.51	3.90	2.7×10^{-5}
	32π	123	3.51	2.63	8.3×10^{-5}
RCM	4π	166	3.48	3.33	8.8×10^{-5}
	8π	195	3.51	4.07	4.3×10^{-5}
	16π	114	3.49	2.53	3.0×10^{-5}
	32π	95	3.45	2.30	6.7×10^{-5}
ND	4π	163	3.50	3.69	5.5×10^{-5}
	8π	145	3.48	3.40	3.0×10^{-5}
	16π	115	3.50	2.77	3.1×10^{-5}
	32π	138	3.49	3.11	7.8×10^{-5}
MMD	4π	162	3.49	3.54	5.9×10^{-5}
	8π	143	3.47	3.16	3.6×10^{-5}
	16π	114	3.48	2.75	4.1×10^{-5}
	32π	137	3.49	3.09	5.7×10^{-5}
MLGC	4π	64	3.49	1.63	6.2×10^{-5}
	8π	40	3.48	1.15	3.1×10^{-5}
	16π	33	3.48	1.08	2.7×10^{-5}
	32π	53	3.49	1.57	8.3×10^{-5}

Table 1: Shifted ILUT with different reorderings on the Helmholtz problem. MLGC: SNCR, top-down and two-sided approach with $\beta = \{0.002, 0.003, 0.0055, 0.016\}$ and 2 levels of coarsening.

each preconditioner.

The results from the table indicate that at lower values of the wavenumber, shifted ILUT with natural ordering performs better than shifted ILUT with RCM, ND, or MMD ordering. At higher wavenumbers, when the system is very indefinite, the RCM ordering appears to perform better than the ND, MMD, and natural orderings. However, shifted ILUT with the MLGC ordering is by far the winner. It exhibits superior performance over the other ordering strategies for all the different values of the wavenumber. It requires fewer iterations to converge, and it converges in very little time compared to the rest. We note here that the time to compute each preconditioner (including the time to perform the reordering) is similar (under 1 second) for the different methods, and hence the total time presented in the table gives a good indication of the solution time, and correlates with the number of iterations required for convergence.

7.3 Anisotropic examples

The numerical solutions to the following problems use a restarted GMRES accelerator, coupled with the level-based ILU preconditioner, ILU(k). Here, we assume convergence whenever the ℓ_2-norm of the initial residual is reduced by a relative factor of 10^{10}. The results for the MLGC strategies use 2 steps of coarsening with the two-sided approach for constructing the coarse level graph, and with local reordering by RCM. In order to represent both options, we use the bottom-up approach for the MLGC-SNCR algorithm and the top-down approach for the MLGC-Relax algorithm. For the tests with MLGC-SNCR, we use a tolerance of $\beta = 0.2$ for the candidate set selection at each level. For the MLGC-Relax approach, we perform 3 iterations of the Jacobi relaxation method, and we set β to 0.9. This choice of β generally makes the MLGC-Relax algorithm more selective, and results in a reordered matrix with a large block in the top-left position ($A_{F_\ell F_\ell}$ in Equation 2), which can benefit local reordering.

In the following numerical results, we fix the fill level parameter for ILU(k) to be the same for all levels of the MLGC algorithms. This is done to offer a fair comparison with the traditional ordering techniques.

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Precon time	Iteration time	Error
Natural(2)	2.68	166	0.02	0.23	1.0×10^{-6}
Natural(3)	3.80	130	0.04	0.21	7.04×10^{-7}
RCM(2)	1.77	118	0.02	0.15	6.28×10^{-7}
RCM(3)	2.29	87	0.02	0.12	4.16×10^{-7}
ND(2)	2.12	172	0.06	0.22	8.96×10^{-7}
ND(3)	2.59	134	0.06	0.18	6.79×10^{-7}
MMD(2)	2.09	175	0.05	0.22	1.0×10^{-6}
MMD(3)	2.53	132	0.04	0.17	5.73×10^{-7}
MLGC-SNCR(2)	2.04	64	0.03	0.08	2.13×10^{-7}
MLGC-SNCR(3)	2.48	39	0.03	0.05	1.88×10^{-7}
MLGC-Relax(2)	1.77	90	0.02	0.12	4.95×10^{-7}
MLGC-Relax(3)	2.29	68	0.03	0.09	3.38×10^{-7}

Table 2: Summary of results for stretched circle problem, $h = 0.02$. MLGC-SNCR: bottom-up, two-sided approach with $\beta = 0.2$; MLGC-Relax: top-down, two-sided approach with $\beta = 0.9$ and 3 Jacobi iterations

7.3.1 Stretched circle problem

Table 2 gives a summary of the results for the stretched circle problem, with a mesh size of 0.02, using ILU(k) with $k = (1, 2, 3)$. For all the different reordering methods, increasing the fill level parameter yields a more accurate factorization and results in better solver convergence. We observe that compared to the natural, MMD, and ND orderings, RCM performs better in terms of the iteration count to convergence, memory efficiency, and computational time. However, the MLGC algorithms outperform any of the standard ordering strategies. For the same cost in memory, the MLGC-Relax algorithm cuts the number of iterations by over 20% and time to solution by over $\approx 15\%$, compared to RCM. The MLGC-SNCR algorithm is slightly more expensive in terms of memory when compared to RCM or the MLGC-Relax algorithm. However, it has the best performance in terms of iteration count and time to solution, with an improvement of over 45% compared to RCM. The results also show that the natural ordering performs better than MMD or ND, although the natural ordering generates more fill-ins, and thus, has a much larger fill factor.

7.3.2 Unstructured Triangles problem

Table 3 shows results for the unstructured triangle problem. For each of the reordering strategies, a fill level parameter, $k = (1, 2)$, is used to construct the ILU preconditioner. The results indicate that the natural ordering yields the least effective preconditioner, while it generates significant fill-ins compared to the other strategies. Once again, RCM outperforms ND and MMD. However, the MLGC strategies show better convergence in terms of iteration count and solution error compared to RCM, with the MLGC-SNCR strategy performing the best. Here, the MLGC-SNCR algorithm is also the most effective at reducing fill-ins generated during the ILU(k) factorization. Notice, however, that the RCM method appears to have a slightly better overall computational time. This is to be expected considering the additional cost incurred by performing local reordering for the MLGC strategies, for a relatively modest improvement in the number of iterations required to reach convergence.

7.3.3 Rotated anisotropy problem

Tables 4 and 5 give the results for the rotated anisotropy problem, with a rotation through an angle of $\pi/16$ and $2\pi/16$ respectively. For these results, a fill level parameter, $k = (1, 2)$, is used to construct the ILU preconditioner. Once again, the results indicate that the natural ordering generates significantly more fill-ins compared to the other strategies. Furthermore, when ILU(1) is used, the natural ordering fails for both test cases. When ILU(2) is used instead, the natural ordering converges in the fewest number of iterations, but at a much higher cost than the other strategies. For the $\pi/16$ problem, the performance of MLGC-SNCR is only slightly better than that of RCM in terms of iteration count and solution error. However, RCM

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Precon time	Iteration time	Error
Natural(1)	4.58	80	3.76	6.40	1.22×10^{-4}
Natural(2)	9.08	56	12.75	6.39	1.20×10^{-4}
RCM(1)	2.06	67	1.48	3.30	1.47×10^{-4}
RCM(2)	3.69	49	3.10	2.79	1.30×10^{-4}
ND(1)	2.15	79	5.31	4.27	1.29×10^{-4}
ND(2)	3.94	56	7.75	3.55	1.21×10^{-4}
MMD(1)	2.21	77	2.89	5.31	1.15×10^{-4}
MMD(2)	3.81	55	5.31	4.29	9.50×10^{-5}
MLGC-SNCR(1)	2.02	64	1.69	3.45	7.75×10^{-5}
MLGC-SNCR(2)	3.60	45	3.46	2.76	7.64×10^{-5}
MLGC-Relax(1)	2.06	66	1.87	3.24	1.25×10^{-4}
MLGC-Relax(2)	3.67	47	3.53	2.64	9.47×10^{-5}

Table 3: Summary of results for unstructured triangle problem MLGC-SNCR: bottom-up, two-sided approach with $\beta = 0.2$; MLGC-Relax: top-down, two-sided approach with $\beta = 0.9$ and 3 Jacobi iterations

has a cheaper memory cost, which leads to a faster time to solution particularly for the ILU(2) case. A similar observation can be made for the $2\pi/16$ problem, although the difference in the iteration count to reach convergence is increased in favor of the MLGC-SNCR algorithmm, particularly for the ILU(1) case. For both examples, the MLGC-Relax algorithm shows worse performance compared to RCM for roughly the same cost in memory. This could be attributed to the poor convergence of the Jacobi relaxation method on this problem. Using the Gauss Seidel relaxation method instead, and with the same parameters, the MLGC-Relax algorithm gives results similar to that of RCM, see Table 6. This is not surprising since the choice of β selects the block sizes of the reordered matrix in favor of the local reordering.

7.4 Effect of variable levels of fill

In the results from the previous sections, we have used a single fill level parameter for the MLGC schemes, in order to allow a fair comparison with traditional ordering techniques. In the following example, we consider the benefit of using a level-of-fill parameter that varies with the levels of the MLGC scheme.

Fill-reducing strategies such as RCM are known to be effective at higher fill levels, since the resulting ILU factorization becomes more accurate. The results in Table 2 indicate that even at low fill levels the MLGC strategies show good convergence. This is a desirable property and we explore this further by considering the more challenging stretched circle problem (Problem II) with a mesh size of 0.01. Due to the smaller mesh size and the strong anisotropy, the problem is more challenging to solve, and a relatively high fill level is needed for the iterative solver to converge. Table 7 shows the results for RCM and MLGC. The MLGC algorithm uses the SNCR strategy with the two-sided approach, bottom-up, and $\beta = 0.2$. Here, we consider the use of both a fixed and a variable level-of-fill parameter k, for the MLGC algorithm. In order to differentiate between the two, we use a subscript v to identify the MLGC algorithm with a variable k (MLGC$_v$). We note here that solutions with ND, MMD, and the natural orderings failed to converge in the required number of iterations, and with a reasonable fill factor. RCM(4) also failed to converge and the convergence of RCM(5) is quite slow. In contrast, MLGC(5) converges about 5 times faster for roughly the same cost in memory usage. In general, the MLGC algorithms show good convergence even at low values of the fill level parameter. Comparing the MLGC strategies further shows the benefit of using a variable k. For roughly the same iteration count, the MLGC$_v$ variant has a cheaper memory cost, compared to the MLGC algorithm with a fixed k. Moreover, in practice, using a variable fill level parameter leads to a more robust strategy since it provides better control over the accumulation of fill-ins at the different levels of the multilevel structure. We note here that the MLGC algorithm with a fixed k did not converge for k less than 3 for this problem.

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Precon time	Iteration time	Error
Natural(1)	3.30	F	1.24	18.89	5.10×10^{2}
Natural(2)	7.08	7	5.19	0.54	2.19×10^{-8}
RCM(1)	1.34	13	0.48	0.33	1.54×10^{-7}
RCM(2)	1.84	10	0.66	0.27	1.45×10^{-7}
ND(1)	1.70	43	2.78	1.50	8.58×10^{-8}
ND(2)	2.30	28	3.30	1.02	7.14×10^{-8}
MMD(1)	1.75	55	0.84	2.05	1.05×10^{-7}
MMD(2)	2.08	27	1.14	0.95	8.14×10^{-8}
MLGC-SNCR(1)	1.87	10	0.81	0.31	1.11×10^{-7}
MLGC-SNCR(2)	2.96	8	1.38	0.29	2.15×10^{-8}
MLGC-Relax(1)	1.34	41	0.82	1.28	1.78×10^{-7}
MLGC-Relax(2)	1.84	28	1.00	0.90	1.55×10^{-7}

Table 4: Summary of results for rotated anisotropic problem rotated by $\pi/16$ MLGC-SNCR: bottom-up, two-sided approach with $\beta = 0.2$; MLGC-Relax: top-down, two-sided approach with $\beta = 0.9$ and 3 Jacobi iterations

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Precon time	Iteration time	Error
Natural(1)	3.30	F	1.24	16.47	6.0×10^{2}
Natural(2)	7.08	6	5.19	0.46	2.02×10^{-7}
RCM(1)	1.34	31	0.48	0.89	2.99×10^{-7}
RCM(2)	1.84	12	0.65	0.34	4.78×10^{-7}
ND(1)	1.70	55	2.83	2.05	2.52×10^{-7}
ND(2)	2.30	39	3.30	1.50	1.69×10^{-7}
MMD(1)	1.75	56	0.84	2.11	2.32×10^{-7}
MMD(2)	2.08	39	1.14	1.43	2.12×10^{-7}
MLGC-SNCR(1)	1.87	13	0.79	0.40	8.02×10^{-8}
MLGC-SNCR(2)	2.96	8	1.37	0.30	5.03×10^{-8}
MLGC-Relax(1)	1.35	120	0.81	4.69	4.87×10^{-7}
MLGC-Relax(2)	1.84	26	0.99	0.82	3.48×10^{-7}

Table 5: Summary of results for rotated anisotropic problem rotated by $2\pi/16$ MLGC-SNCR: bottom-up, two-sided approach with $\beta = 0.2$; MLGC-Relax: top-down, two-sided approach with $\beta = 0.9$ and 3 Jacobi iterations

Problem	ILU(k)	Fill factor	Iteration count	Precon time	Iteration time	Error
Problem IV, $\theta = \frac{\pi}{16}$	ILU(1)	1.34	17	0.82	0.45	4.42×10^{-8}
	ILU(2)	1.84	13	0.96	0.37	4.25×10^{-8}
Problem IV, $\theta = \frac{2\pi}{16}$	ILU(1)	1.34	34	0.80	0.99	1.87×10^{-7}
	ILU(2)	1.84	14	1.00	0.40	1.20×10^{-7}

Table 6: MLGC-relax on Problem IV, $\theta = \frac{\pi}{16}$ and $\theta = \frac{2\pi}{16}$, using the top-down, two-sided approach with $\beta = 0.9$ and 3 Gauss Seidel iterations

8 Summary and Conclusion

8.1 Summary

The numerical results presented above show an improvement in performance when the MLGC strategy is used, compared to the natural and other standard reordering strategies. One explanation for this comes

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Total time	Error
RCM(4)	2.89	F	2.32	1.31×10^{2}
RCM(5)	3.49	277	2.22	2.62×10^{-5}
MLGC(3)	2.52	141	1.11	8.36×10^{-6}
MLGC(4)	3.00	76	0.71	1.42×10^{-6}
MLGC(5)	3.50	57	0.79	1.88×10^{-6}
MLGC$_v$(2)	2.89	79	0.72	1.83×10^{-6}
MLGC$_v$(3)	3.39	59	0.59	9.64×10^{-7}

Table 7: Summary of results for MLGC and RCM for different levels of fill on the stretched circle problem, $h = 0.01$. RCM(k) (resp. MLGC(k)) denotes an ILU(k) factorization is performed on the RCM (resp. MLGC) reordered system matrix. The MLGC algorithm using a variable fill level parameter is denoted MLGC$_v$. The MLGC algorithm uses MLGC-SNCR: bottom-up, two-sided approach and $\beta = 0.2$

from considering the effect of the MLGC scheme on the stability and accuracy of the ILU factorization. First, recall that the coarsening strategy compares a node to its largest neighbor (based on some weight), and assigns one to the coarse set and the other to the fine set. As a result, long recurrences during the preconditioner solve (i.e. forward solve with L and backward solve with U) involving large (neighboring) entries in L and U are potentially avoided. In other words, the norm of L^{-1} or U^{-1} is not too large, leading to more stable triangular solves. A theoretical basis of this result for the symmetric positive definite case is presented in [18]. Furthermore, the splitting of nodes into coarse or fine sets leads to a hierarchy of (block) independent sets, which can be eliminated with very little fill-ins during the factorization. Thus, few entries are dropped during the factorization, which suggests that the preconditioner obtained from the ILU factorization with this ordering should be more accurate. Moreover, since nodes that are poor in terms of diagonal dominance, (or nodes with unfavorable pivots), are relegated to be ordered last, the adverse effects of these poor nodes on the factorization is minimized. The result is that, large entries in L or U, which could lead to unstable and inaccurate factors, are avoided. We note here that for the indefinite cases, such as the Helmholtz problem at high wavenumbers, the use of diagonal compensation or modification strategies play a major role in bounding the norm of L^{-1} or U^{-1}, leading to stable triangular solves (see for instance [45, 48, 57]). In such cases, reordering may help to improve performance as shown in Table 1. However, reordering alone may not be sufficient to solve such problems.

Unlike the MLGC scheme, the other reordering techniques (i.e. RCM, ND, and MMD) are based primarily on the adjacency graph of the coefficient matrix A without any consideration given to algebraic values of its entries. Thus, as indicated by the results in Table 1, when the natural ordering gives an ordering that is good in terms of minimizing fill-ins, it may not be beneficial to reorder the system by an approach that is based solely on its graph.

8.2 Conclusion

The technique presented in this paper exploits a multilevel graph coarsening strategy to define an effective reordering for ILU-based preconditioners. The direction of coarsening for the multilevel method follows either a top-down or bottom-up approach, and two different strategies are presented for defining the splitting of nodes into coarse and fine sets. The first strategy considers the strength of connection between neighboring nodes, and the second is based on the use of relaxation techniques from algebraic multigrid. With the help of examples from the solution of the Helmholtz equation, and anisotropic problems from 2D/3D PDEs, we compared the results obtained with the proposed technique against those obtained from the natural ordering, as well as other standard reordering techniques, namely RCM, MMD and ND.

For the Helmholtz problem, the results indicate that the multilevel graph coarsening reordering strategy is superior to the natural ordering, or the ordering given by RCM, ND, or MMD. When used in combination with shifted ILUT, significant gains are observed in the number of iterations required to reach convergence, and in the overall computational time, even for the highly indefinite cases. At high wavenumbers, reordering alone is not enough to efficiently solve the Helmholtz problem and robust algebraic strategies such as diagonal

compensation (Shifted ILU) or modified ILU are necessary to ensure good convergence.

The results from the anisotropic examples also indicate significant performance gains over the natural ordering. In general, comparative results show that the MLGC approach either outperformed the standard reordering strategies or gave results comparable to RCM, which in this case was the best of the standard reordering schemes.

Our results also indicate that performing some local reordering within the blocks of the MLGC scheme can be beneficial. In practice, any reordering strategy may be utilized to reorder the local blocks. In this work, we used RCM to perform local reordering for the anisotropic examples, in order to benefit from its fill-reducing property. Depending on the tolerance used in the preselection phase of the MLGC algorithm, one can control the size of the local blocks, and hence, the resulting fill factor of the ILU factorization.

To summarize, we observe that when the natural ordering reduces fill-ins, it generally performs quite well compared to the standard reordering strategies. Furthermore, reorderings based solely on graph theory may be ineffective for highly indefinite problems, such as the Helmholtz problem at high wave numbers, and problems with strong (unstructured) anisotropy. In such cases, combining ideas from graph theory with algebraic information from the underlying adjacency matrix can be quite beneficial. These observations are in agreement with previous results from the literature, see e.g., [18, 19, 27]. Due to its algebraic nature, the MLGC strategy requires some parameter tuning, which makes it applicable to a wider range of problems but at the expense of added complexity. Furthermore, one can easily adapt the framework to a specific problem by defining a new C/F splitting strategy that is appropriate for that problem.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. We wish to thank Kechroud, Gowda and Soulaimani for providing us with the finite-element code for acoustic wave scattering (see [43]), and J. Schroder for providing us with the data for the anisotropic examples presented in the experiments.

References

[1] *MFEM: Modular parallel finite element methods library.* http://mfem.googlecode.com.

[2] P. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 886–905.

[3] O. Axelsson, *A generalized SSOR method*, BIT, 12 (1972), pp. 443–467.

[4] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods. I*, Numer. Math., 56 (1989), pp. 157–177.

[5] R. E. Bank, *Compatible coarsening in the multigraph algorithm*, Advances in Engineering Software, 38 (2007), pp. 287–294.

[6] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numerische Mathematik, 82 (1999), pp. 543–576.

[7] M. Benzi, J. C. Haws, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, Tech. Report LA-UR-99-4857, CIC-19 and Los Alamos National Laboratory, Los Alamos, NM 87545, August 1999.

[8] M. Benzi, D. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1652–1670.

[9] M. Bollhöfer, *A robust ILU with pivoting based on monitoring the growth of the inverse factors*, Linear Algebra and its Applications, 338 (2001), pp. 201–213.

[10] M. Bollhöfer, M. J. Grote, and O. Schenk, *Algebraic multilevel preconditioner for the helmholtz equation in heterogeneous media*, SIAM Journal on Scientific Computing, 31 (2009), pp. 3781–3805.

[11] E. Botta, A. Ploeg, and F. Wubs, *Nested grids ILU-decomposition (NGILU)*, J. Comp. Appl. Math., 66 (1996), pp. 515–526.

[12] E. Botta and F. Wubs, *Matrix Renumbering ILU: an effective algebraic multilevel ILU*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 1007–1026.

[13] A. Brandt, *Generally highly accurate algebraic coarsening*, Electronic Transactions on Numerical Analysis, 10 (2000), pp. 1–20.

[14] A. Brandt, S. F. Mc Cormick, and J. Ruge, *Algebraic multigrid (amg) for sparse matrix equations*, in Sparsity and its applications, D. J. Evans, ed., Cambridge, 1984, Cambridge Univ. Press.

[15] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *An energy-based AMG coarsening strategy*, Numerical Linear Algebra with Applications, 13 (2006), pp. 133–148.

[16] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov, *Algebraic multilevel preconditioners for the graph Laplacian based on matching in graphs*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 1805–1827.

[17] J. Brannick and R. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM Journal on Scientific Computing, 32 (2010), pp. 1393–1416.

[18] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored approximate inverses*, SIAM Journal on Scientific Computing, 21 (1999), p. 867882.

[19] ——, *A structural diagnosis of some ic orderings*, SIAM Journal on Scientific Computing, 22 (2000), p. 15271532.

[20] W. L. Briggs, V. E. Henson, and S. F. Mc Cormick, *A multigrid tutorial*, SIAM, Philadelphia, PA, 2000. Second edition.

[21] A. M. Bruaset, A. Tveito, and R. Winther, *On the stability of relaxed incomplete LU factorizations*, Mathematics of Computation, 54 (1990), pp. 701–719.

[22] N. Buleev, *A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion*, Math. Sb., 51 (1960), pp. 227–238.

[23] T. Chan and H. Elman, *Fourier analysis of iterative methods for elliptic problems*, SIAM Review, 31 (1989), pp. 20–49.

[24] J. Chen and I. Safro, *Algebraic distance on graphs*, SIAM Journal on Scientific Computing, 33 (2011), pp. 3468–3490.

[25] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, UK,, 2005.

[26] E. Chow and P. S. Vassilevski, *Multilevel block factorizations in generalized hierarchical bases*, Numerical Linear Algebra with Applications, 1 (2002), pp. 1–22.

[27] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), p. 944961.

[28] S. Doi and A. Hoshi, *Large numbered multicolor MILU preconditioning on SX-3/14*, Int'l J. Computer Math., 44 (1992), pp. 143–152.

[29] I. Duff and G. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT Numerical Mathematics, 29 (1989), pp. 635–657.

[30] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, New York, 1986.

[31] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM Journal on Numerical Analysis, 22 (2001), pp. 973–996.

[32] T. Dupont, R. Kendall, and H. Rachford, *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559–573.

[33] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Mathematics of Computation, 47 (1986), pp. 191–217.

[34] R. Falgout and P. Vassilevski, *On generalizing the AMG framework*, SIAM Journal on Numerical Analysis, 42 (2005), pp. 1669–1693.

[35] A. George, *Nested dissection of a regular finite element mesh*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 345–363.

[36] A. George and J. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[37] ——, *Evolution of the minimum degree ordering algorithm*, SIAM review, 31 (1989), pp. 1–19.

[38] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[39] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1985.

[40] Y. F. Hu and R. J. Blake, *Load balancing for unstructured mesh applications*, Nova Science Publishers, Inc., Commack, NY, USA, 2001, pp. 117–148.

[41] G. Karypis and V. Kumar, *Multilevel graph partitioning schemes*, in Proc. 24th Intern. Conf. Par. Proc., III, CRC Press, 1995, pp. 113–122.

[42] G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and computing Fill-Reducing Orderings of Sparse Matrices, version 4.0.1,University of Minnesota, Dept. of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis*, 1998.

[43] R. Kechroud, A. Soulaimani, Y. Saad, and S. Gowda, *Preconditioning techniques for the solution of the Helmholtz equation by the finite element method*, Math. Comput. Simul., 65 (2004), pp. 303–321.

[44] O. Livne, *Coarsening by compatible relaxation*, Numerical Linear Algebra with Applications, 11 (2004), pp. 205–227.

[45] S. Maclachlan, D. Osei-Kuffuor, and Y. Saad, *Modification and compensation strategies for threshold-based incomplete factorizations*, sisc, 34 (2012), pp. 48–75.

[46] S. MacLachlan and Y. Saad, *Greedy coarsening strategies for nonsymmetric problems*, SIAM J. Sci. Comput., 29 (2007), pp. 2115–2143.

[47] S. MacLachlan and Y. Saad, *A greedy strategy for coarse-grid selection*, SIAM Journal on Scientific Computing, 29 (2007), pp. 1825–1853.

[48] M. Magolu Monga Made, R. Beauwens, and G. Warzee, *Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix*, Comm. in Numer. Meth. in Engin., 16 (2000), pp. 801–817.

[49] T. A. Manteuffel, *Shifted incomplete Cholesky factorization*, in Sparse Matrix Proceedings 1978 (Sympos. Sparse Matrix Comput., Knoxville, Tenn., 1978), SIAM, Philadelphia, Pa., 1979, pp. 41–61.

[50] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[51] N. Munksgaard, *Solving sparse symmetric sets of linear equations by preconditioned conjugate gradient method*, ACM Transactions on Mathematical Software, 6 (1980), pp. 206–219.

[52] A. C. Muresan and Y. Notay, *Analysis of aggregation-based multigrid*, SIAM Journal on Scientific Computing, 30 (2008), pp. 1082–1103.

[53] Y. Notay, *DRIC: a dynamic version of the RIC method*, Numer. Lin. Alg. Applic., (1994). to appear.

[54] ——, *An aggregation-based algebraic multigrid method*, Electronic Transactions on Numerical Analysis, 37 (2010), pp. 123–146.

[55] T. Oliphant, *An implicit numerical method for solving two-dimensional time-dependent diffusion problems*, Quart. Appl. Math., 19 (1961), pp. 221–229.

[56] M. Olschowka and A. Neumaier, *A new pivoting strategy for gaussian elimination*, Numerical Linear Algebra with Applications, 240 (1996), pp. 131–151.

[57] D. Osei-Kuffuor and Y. Saad, *Preconditioning Helmholtz linear systems*, Applied Numerical Mathematics, 60 (2009), pp. 420–431.

[58] D. Ron, I. Safro, and A. Brandt, *Relaxation-based coarsening and multiscale graph organization*, SIAM Multiscale Modelling and Simulations, 9 (2011), pp. 407–423.

[59] D. J. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, (1972), pp. 183–217.

[60] A. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, 1987, ch. 4.

[61] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific and Statistical Computing, 14 (1993), pp. 461–469.

[62] ——, *ILUT: a dual threshold incomplete ILU factorization*, Numerical Linear Algebra with Applications, 1 (1994), pp. 387–402.

[63] ——, *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM Journal on Scientific Computing, 17 (1996), pp. 830–847.

[64] ——, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelpha, PA, 2003.

[65] ——, *Multilevel ILU with reorderings for diagonal dominance*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1032–1057.

[66] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numerical Linear Algebra with Applications, 9 (2002).

[67] J. Schberl, *NETGEN An advancing front 2D/3D-mesh generator based on abstract rules*, Computing and Visualization in Science, 1 (1997), pp. 41–52.

[68] J. Schöberl, J. Gerstmayr, and R. Gaisbauer, *NETGEN - automatic 3d tetrahedral mesh generator.* http://www.hpfem.jku.at/netgen/, 2003.

[69] J. B. Schroder, *Smoothed aggregation solvers for anisotropic diffusion*, Numerical Linear Algebra with Applications, 19 (2012), pp. 296–312.

[70] W. F. Tinney and J. W. Walker, *Direct solution of sparse network equations by optimally ordered triangular factorization*, Proc. IEEE, 55 (1967), pp. 1801–1809.

[71] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.

[72] H. van der Vorst, *The convergence behaviour of preconditioned CG and CGS in the presence of rounding errors*, in Preconditioned conjugate gradient methods, O. Axelsson and L. Kolotilina, eds., vol. 1457, Lecture notes in Math., Springer Verlag, 1990.

[73] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.

[74] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[75] R. Varga, *Factorization and normalized iterative methods*, in Boundary problems in differential equations, R. Langer, ed., Univ. of Wisconsin Press, Madison, 1960, pp. 121–142.

[76] J. W. Watts III, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Society of Petroleum Engineers Journal, 21 (1981), pp. 345–353.